Description
Intended User
Features
User Interface Mocks
- Screen 1 Login
- Screen 2 Forgot password
- Screen 3 Register client
- Screen 4 Request tortillas
- Screen 5 Delivery Notification
- Screen 6 Bike driver login
- Screen 7 Orders to deliver
- Screen 8 Confirm order as delivered
- Screen 9 Branch Current Stock
- Screen 10 Branch login
- Screen 11 Bike drivers
- Screen 12 Register bike driver
- Screen 13 Edit bike driver

Key Considerations
- Architecture
- How will your app handle data persistence?
- Corner cases
- Notes
- Describe any libraries you'll be using and share your reasoning for including them.
- Describe how you will implement Google Play Services.

Next Steps: Required Tasks
- Task 1: Configure central data base
- Task 2: Google cloud endpoint
- Task 3: Client Project Setup
- Task 4: QuieroTortillas
- Task 5: LaBuenaBiker
- Task 6: LaBuena

**GitHub Username**: clemente-morales

# Tortillas requestor

## Description

QuieroTortillas

LaBuena is a small family business to manufacture corn tortillas based on industrial machinery. Tortillas is a staple food in Mexico.

The process to deliver tortillas takes time, effort, and money. Currently we sell tortillas at home and we go from town to town in a truck with a  special sound, so people who wants tortillas comes outside and buy them. Most of the time we spent gas traversing some towns and we don't sell anything. People buying tortillas at home have to walk some blocks, and make a line to finally get their tortillas.

Because of arriving of new competitors and a low demand, We are planning to move LaBuena from our town to a bigger city.

This time we want to improve our distribution plan, using mobile technology and bikes.

We want to publish an application for the customer to request their tortillas and based on location, deliver them as fast as possible in the bikes. The bikers need an application to keep track of the places to deliver and to keep track of their stock. LaBuena requires access to the current external stock, so we are able to fulfill demand.

For the first version of the application we will let the user to pay only in cash.

## Intended User

This application is intended for clients who consume tortillas and are located inside a defined area for home delivery.

## Features

The main features of QuieroTortillas client app includes:

- Register as a client. The user has the option to authenticate using Gmail, Facebook or registering to the application using email and password.
- Add amount of tortillas to buy
- Request tortillas.

The main features of the LaBuenaBikeDriver app includes:

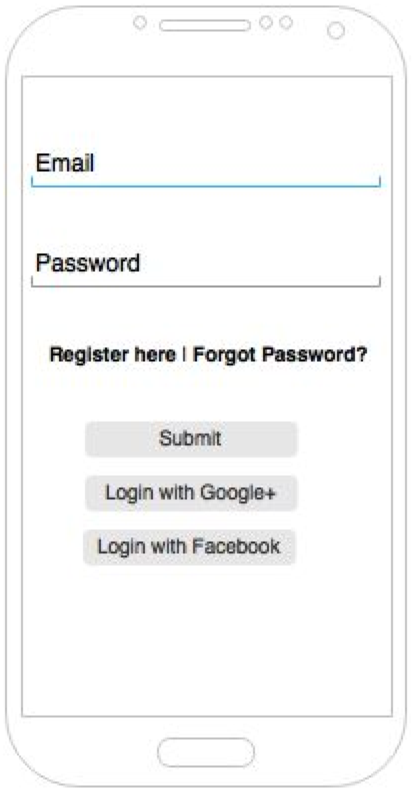
- Display the places and amount of tortillas to deliver
- Send information of the last location every 15 minutes
- Mark an order as paid and delivered. Once marked this data has to be sent to the central server to keep track of the current stock by bike.

- Smart Watch Notification of new order to deliver

The main features of the LaBuena application includes:
- Display of bikes with the latest location and stock
- Information Widget to display pending total orders, external stock, and orders to manufacture

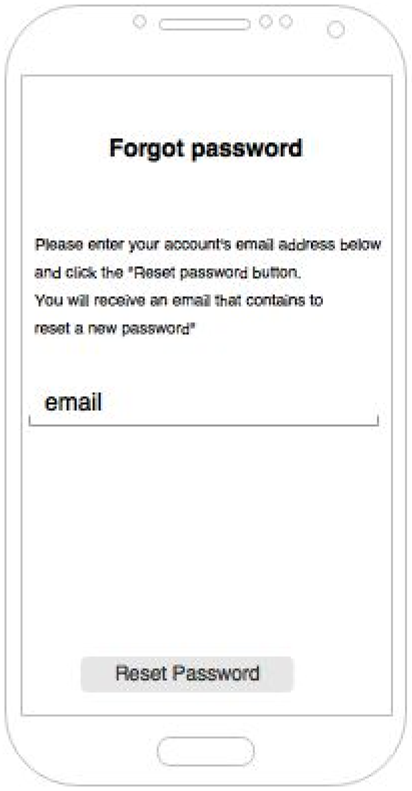
## User Interface Mocks

### Screen 1 Login



This view allows the user to login using an email and password. Clicking register here will allow the user to register as a tortillas consumer client. Clicking in forgot password will take the user to the forgot password view. The user also has the option to login using Google or Facebook.

## Screen 2 Forgot password

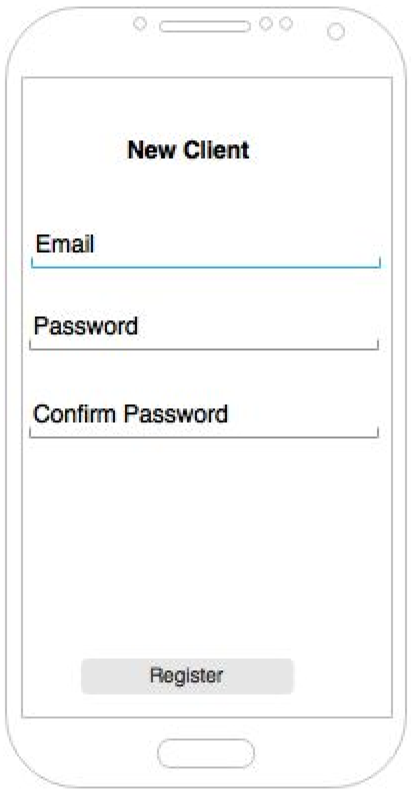

Forgot password

Please enter your account's email address below and click the "Reset password button. You will receive an email that contains to reset a new password"

This view let the user to recover the password using his email. This option only apply if the user register using an email and password in the application.

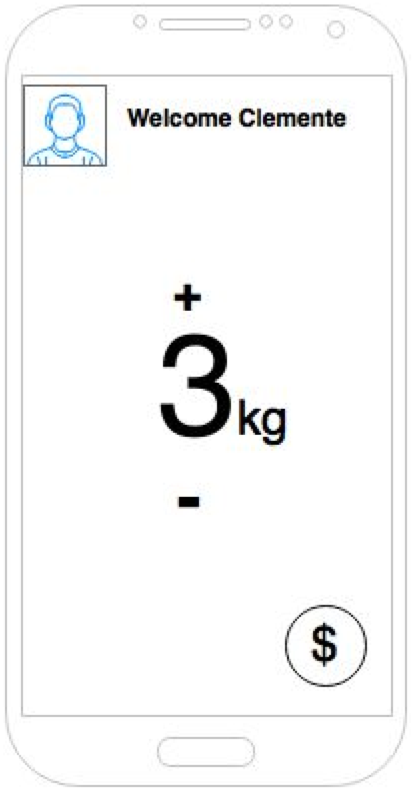
Once the request is send the client will get notification to indicate that a password reset email has been sent to the input email.

## Screen 3 Register client



This view let the user to register as a client. Clicking register will take the client to the address registration view.

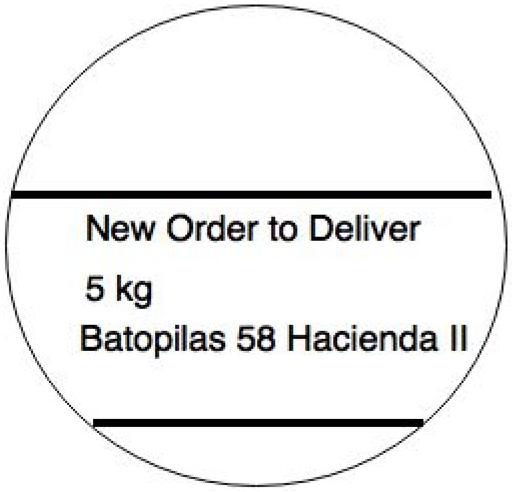
## Screen 4 Request tortillas



This view let the user to request tortillas. Clicking -, decrease the amount to request. Clicking + increase the tortillas amount. Clicking in the $ button will make the tortillas request. Once the order is sent, the user will see a notification for the request made.

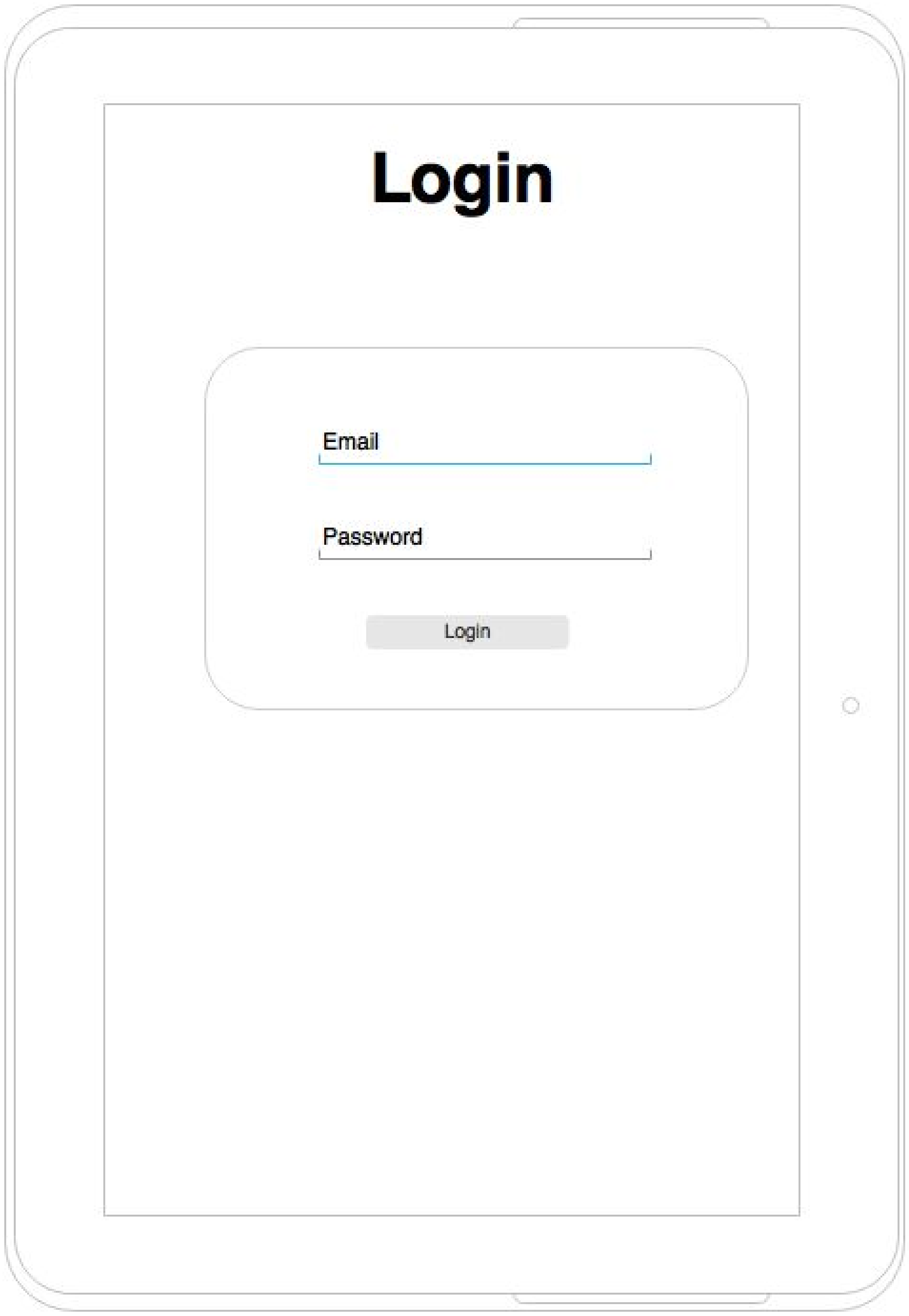
By default the system will take the last known location for the user to make the delivery.

### Screen 5 Delivery Notification



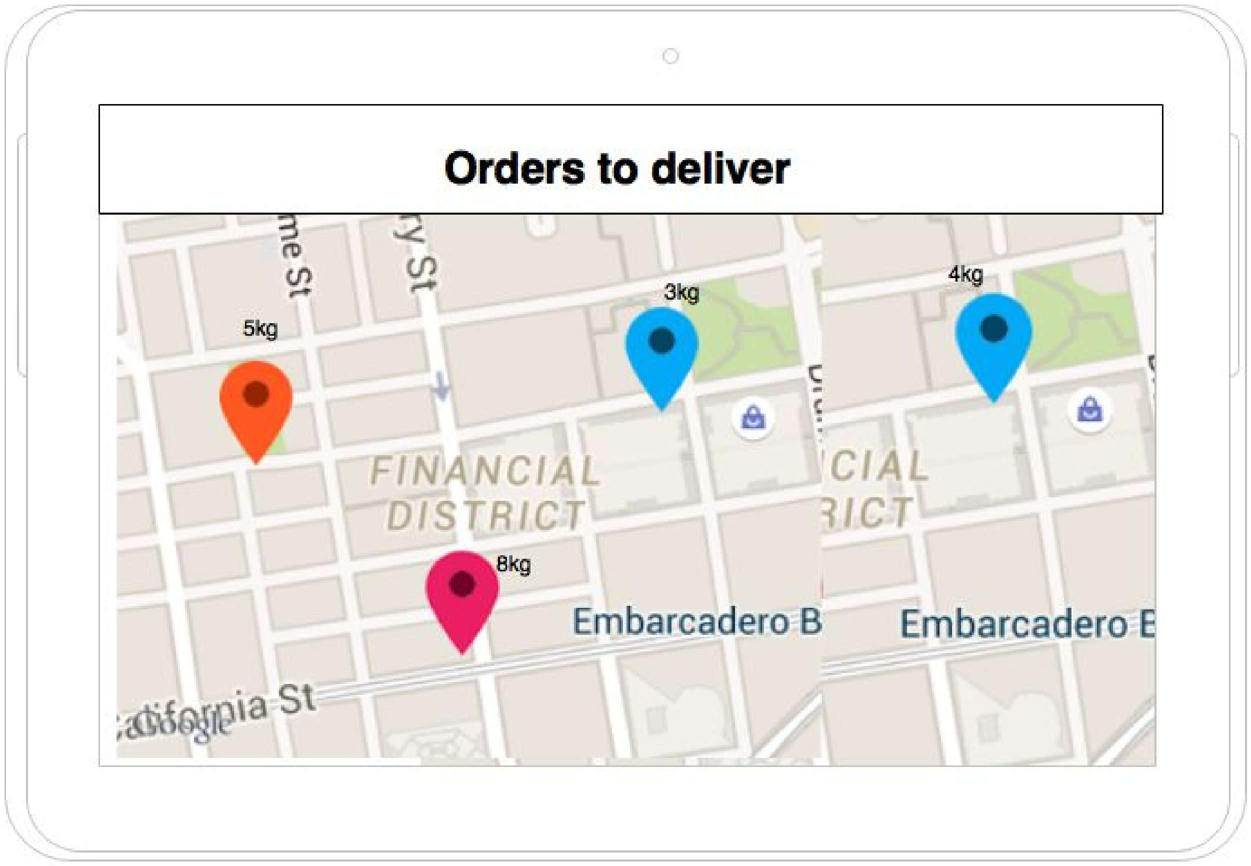
Once a tortillas request has made, the biker will receive a notification in his smart watch to indicate that he has a new delivery. Clicking in the notification will open  the view to display all the pending orders to deliver.

## Screen 6 Bike driver login



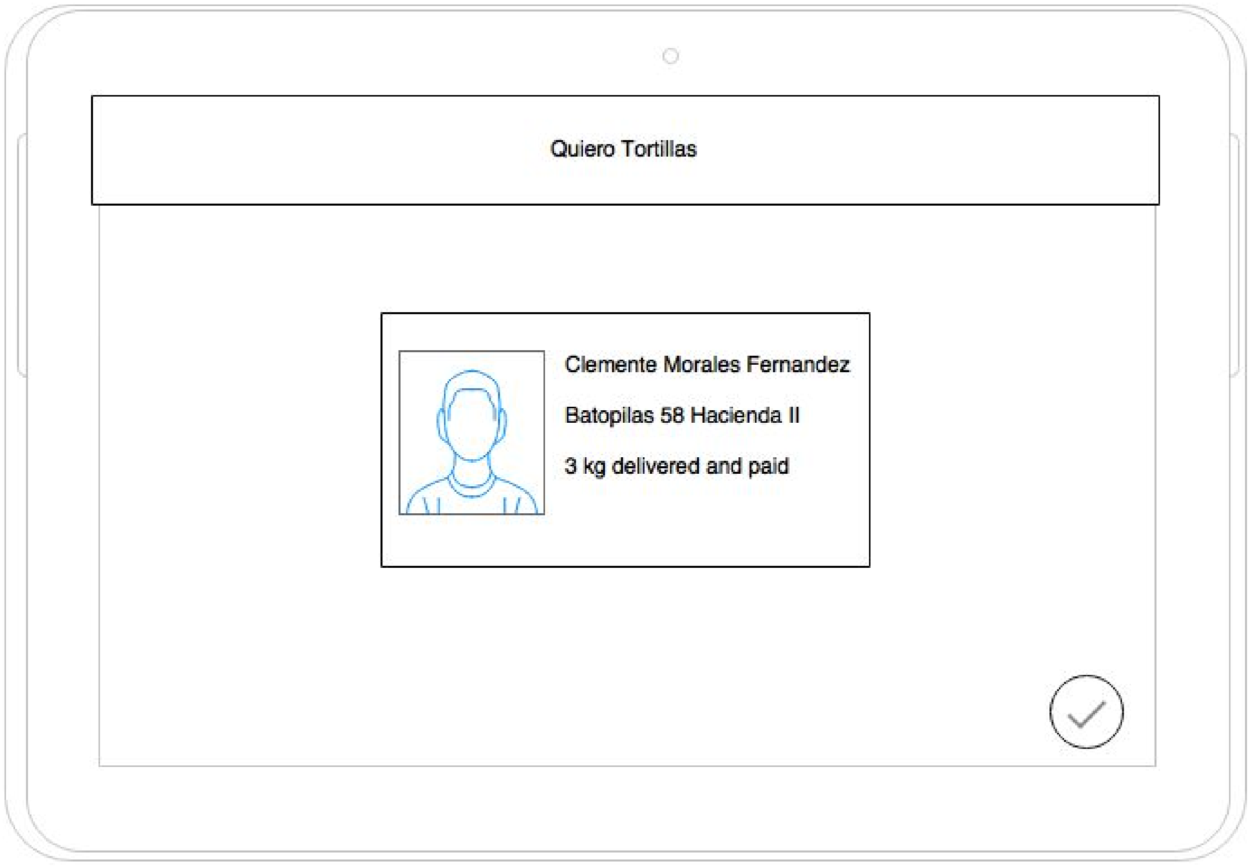
This view let the bike driver to access the application. Once login the application will display the orders to deliver.

## Screen 7 Orders to deliver



The biker will see his pending orders to deliver in a map. Each order will be added as a mark in the maps with the amount to deliver. Clicking in the mark will take the user to the delivery confirmation view.

## Screen 8 Confirm order as delivered



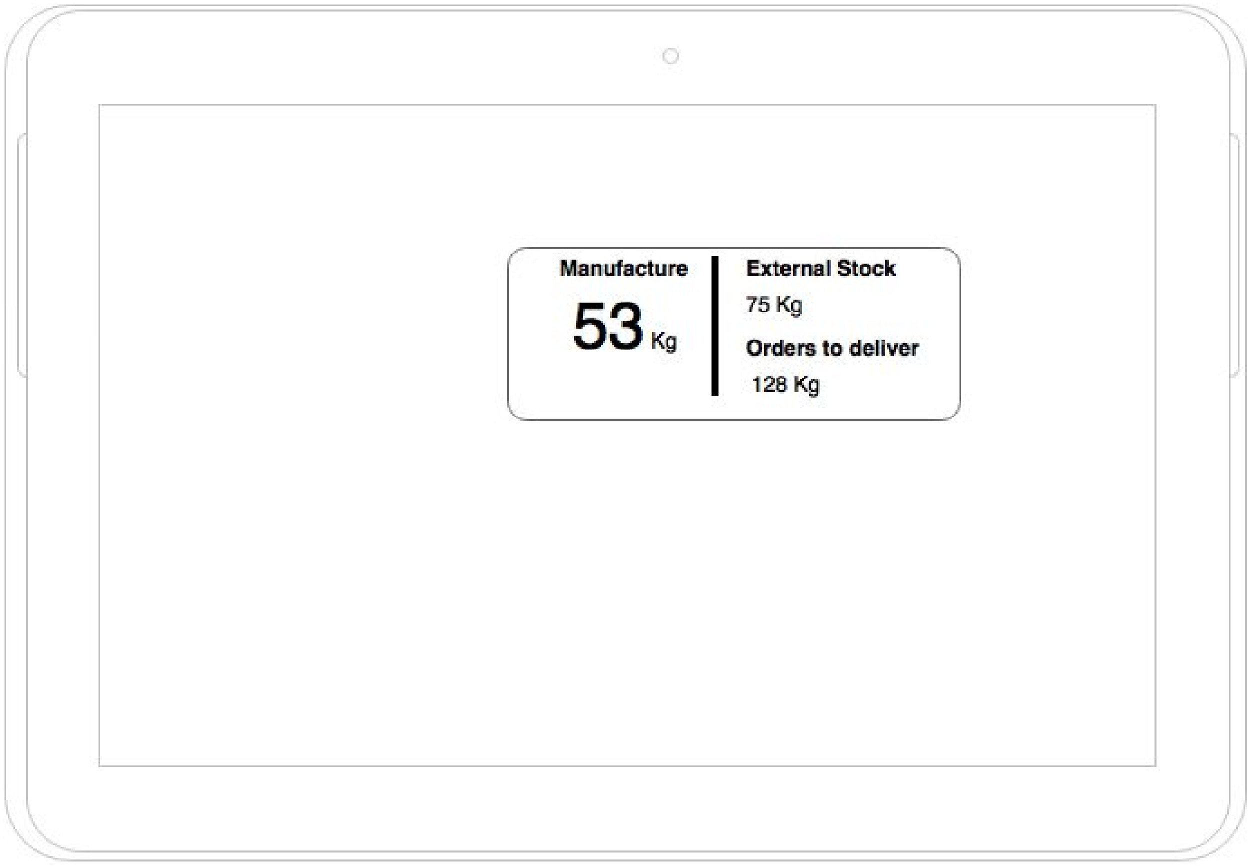
Clemente Morales Fernandez

Batopilas 58 Hacienda II

3 kg delivered and paid

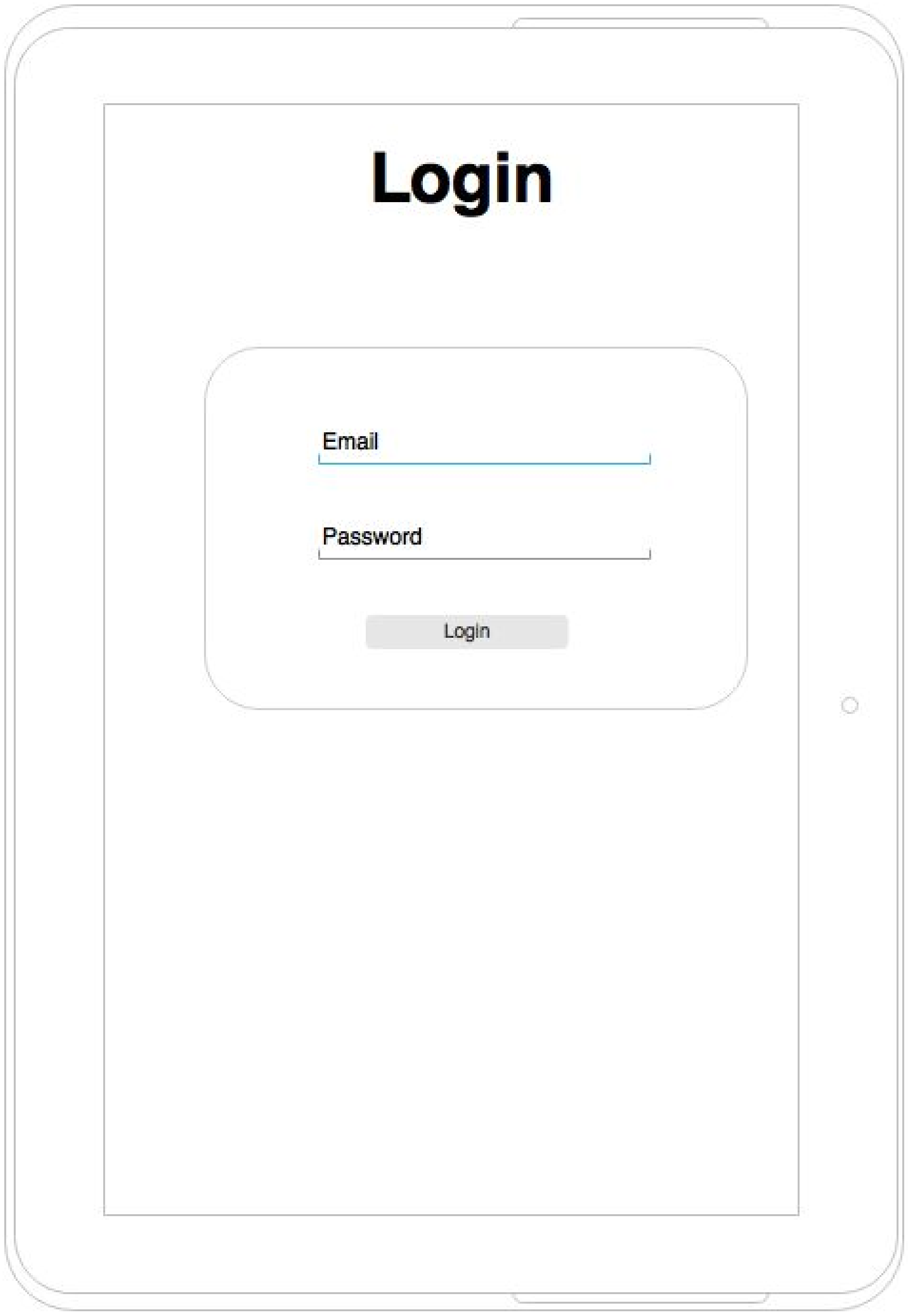
This view will display the detail of the selected order in the map with a button to mark the order as delivered and paid. Once an order is marked as delivered, a request is sent to the server to changed the record in the central data base. Locally we keep track of the order status.

## Screen 9 Branch Current Stock



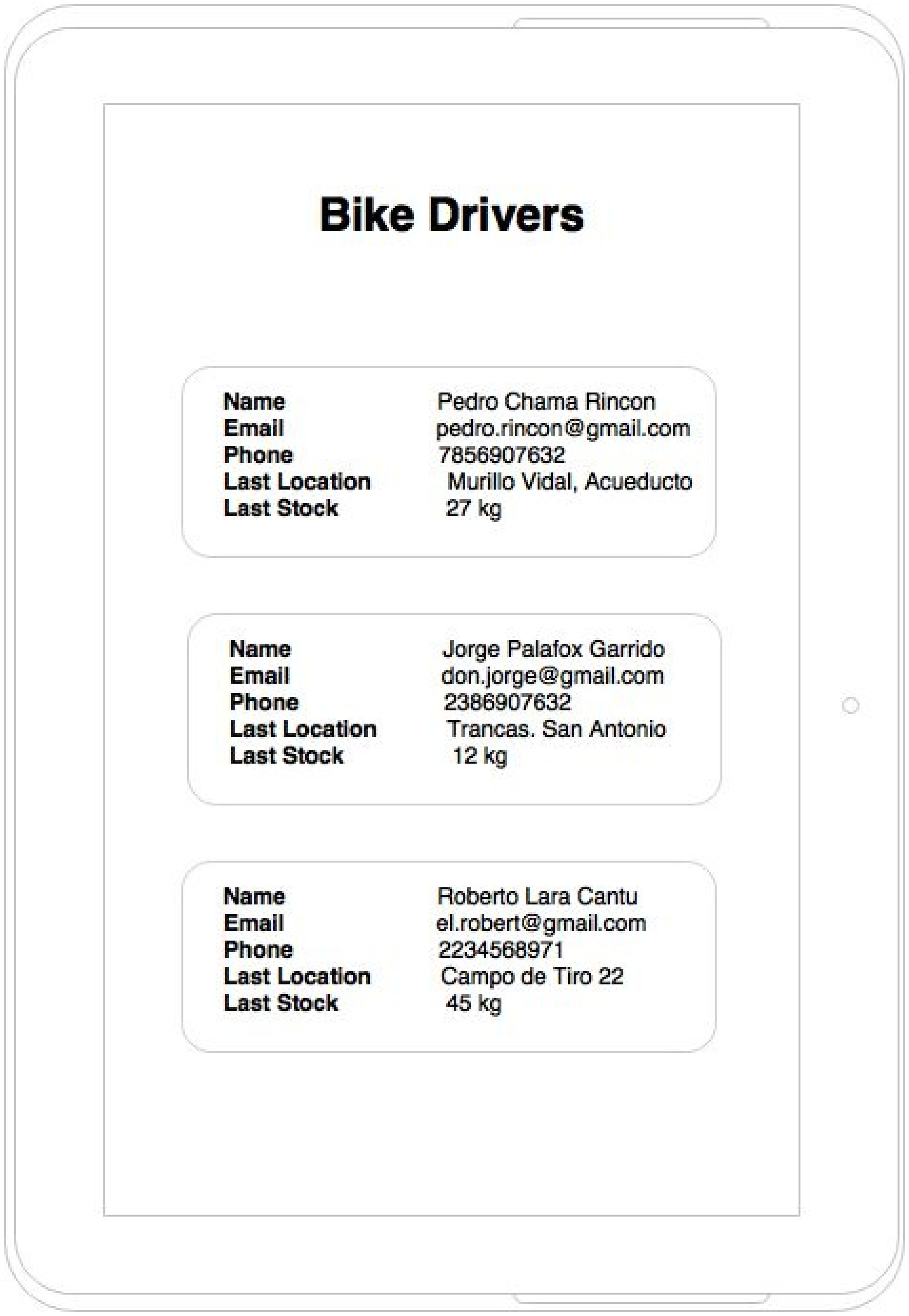
Provides the external stock in the bikes, the orders pending to deliver and the amount of tortillas to manufacture to fulfill the demand.

## Screen 10 Branch login



This view let the branch to access the application. Once login the application will display the bike drivers.

## Screen 11 Bike drivers



Bike Drivers

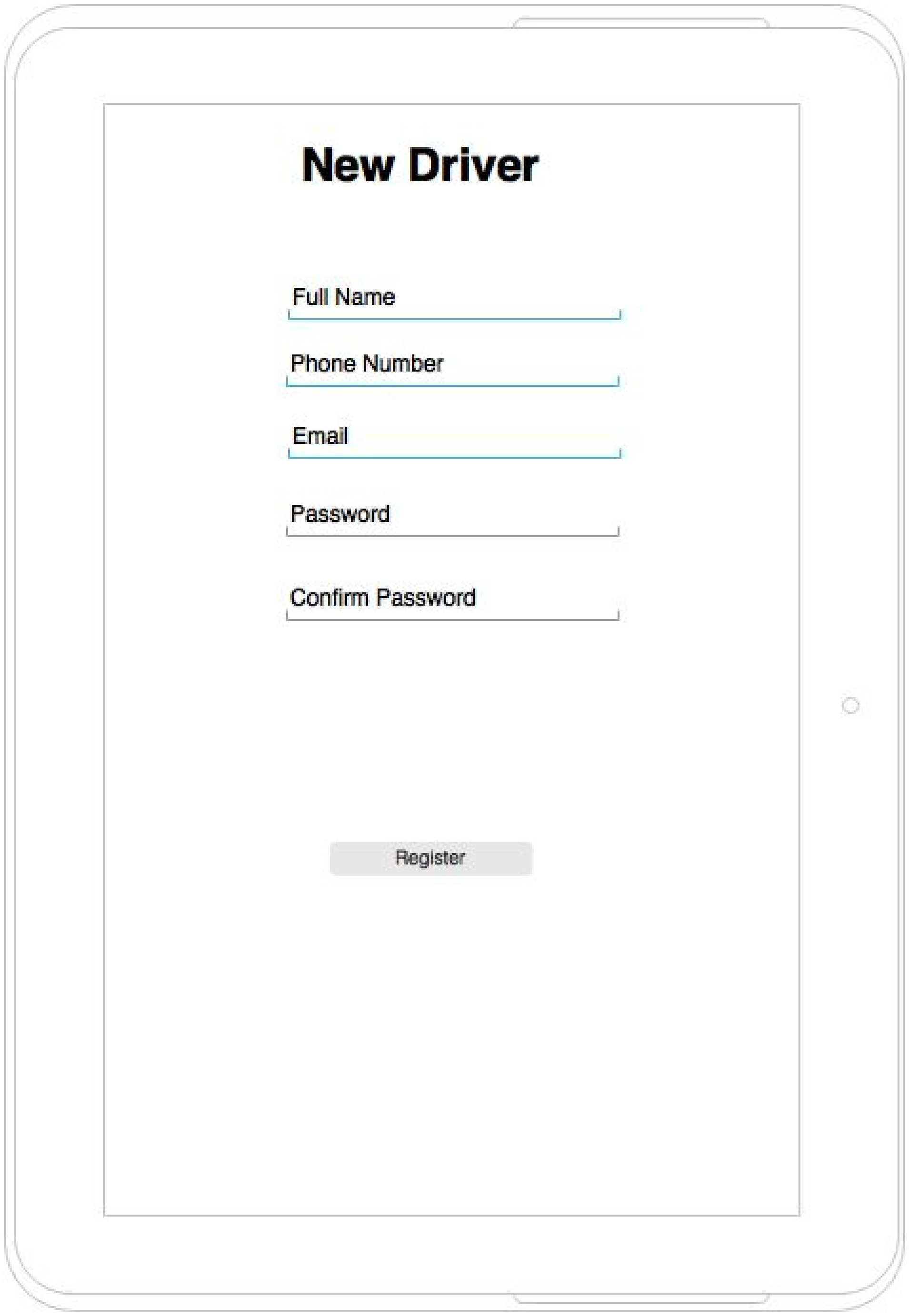
| | |
|---|---|
| **Name** | Pedro Chama Rincon |
| **Email** | pedro.rincon@gmail.com |
| **Phone** | 7856907632 |
| **Last Location** | Murillo Vidal, Acueducto |
| **Last Stock** | 27 kg |

| | |
|---|---|
| **Name** | Jorge Palafox Garrido |
| **Email** | don.jorge@gmail.com |
| **Phone** | 2386907632 |
| **Last Location** | Trancas. San Antonio |
| **Last Stock** | 12 kg |

| | |
|---|---|
| **Name** | Roberto Lara Cantu |
| **Email** | el.robert@gmail.com |
| **Phone** | 2234568971 |
| **Last Location** | Campo de Tiro 22 |
| **Last Stock** | 45 kg |

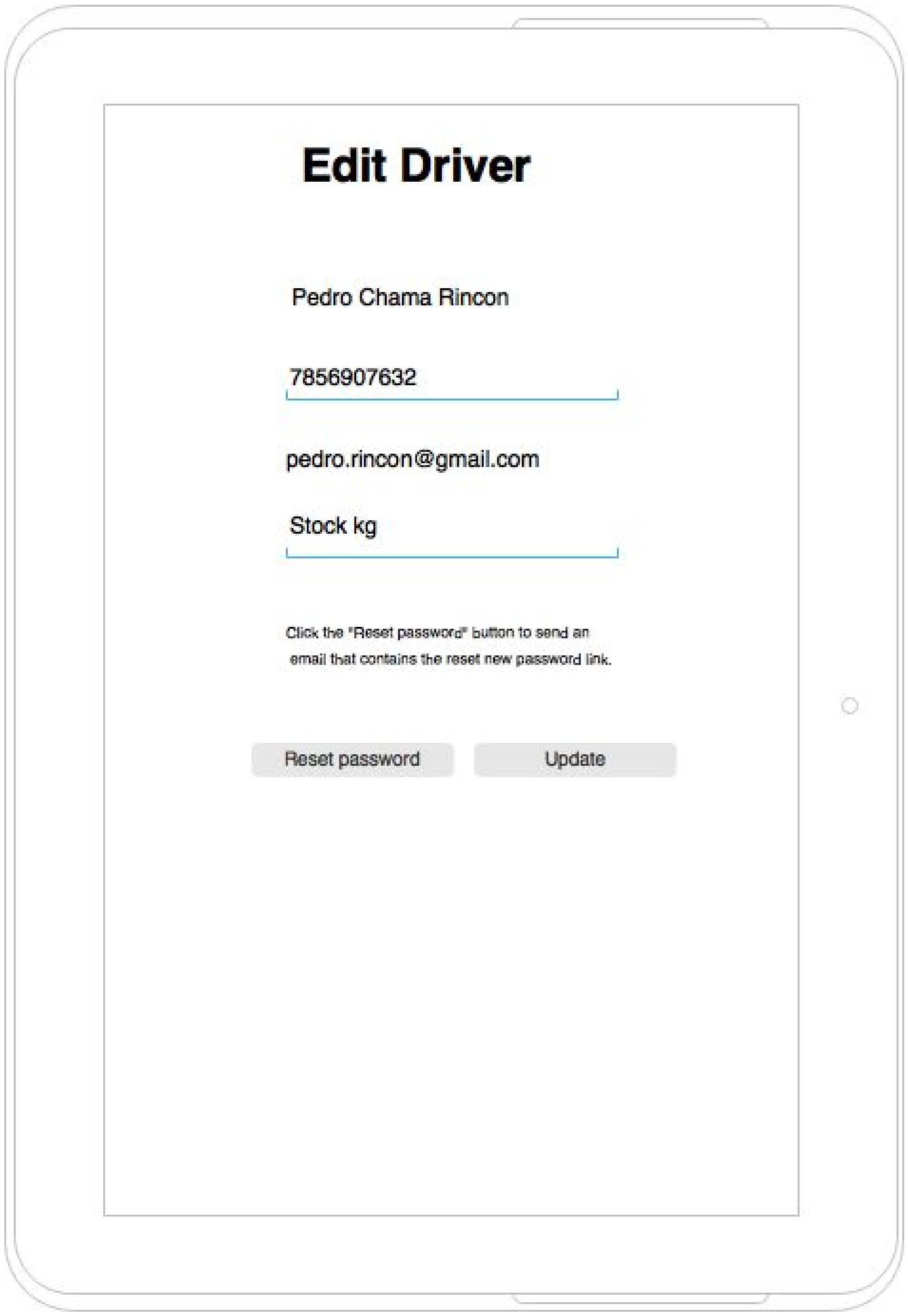
Display the list of register bike drivers. Each bike driver is reporting the last location every 15 minutes, this information is used to know his last location and in a future send the delivery orders based on the location and stock of the driver

## Screen 12 Register bike driver



Allows to register a new bike for the branch.

## Screen 13 Edit bike driver



Edit Driver

Pedro Chama Rincon

7856907632

pedro.rincon@gmail.com

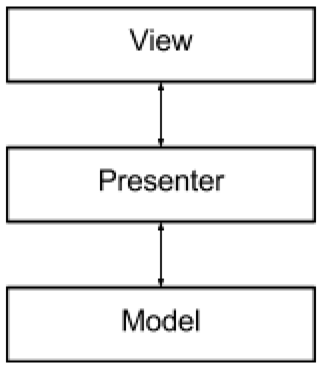
Stock kg

Click the "Reset password" button to send an email that contains the reset new password link.

Let the user to edit the info of  bike driver. The user is able to ask for a reset password email, edit the phone number or add tortillas to the driver stock.

## Key Considerations

**Architecture**
This application will be based on the Model View Presenter pattern.  These style of architecture will make the application easy to maintain and test.



The **View** will handle the display of the data. All the user interaction will be deliver to the presenter.

The **Model** will provide and store data.

The **Presenter** will coordinate the interaction between the View and the Model.

### How will your app handle data persistence?

The persistence of the information will be done by the use of Repositories. This repositories will be injected in the presenters

Repositories will encapsulate required operations to interact with sqlite database

## Corner cases

When the server throws an exception in marking an order as delivered, the application will try to send all the pending orders to synchonize in local data base in the next order to sent to server.

### Notes

Initial support to only one branch of “la buena”
Bike selection is based on the workload assigned to each biker.

### Describe any libraries you’ll be using and share your reasoning for including them.

- **Dagger 2**. For dependency Injection. It will make easy to interact between the presenter and the view.
- **EventBus 3**. To interact between presenter and view. Instead of defining Interfaces in each presenter to communicate with the view, I will use events to notify the the view about data changes.
- **Firebase**. I need to know the identity of the user to bind them with the tortillas orders. Because the business core of LaBuena is not authentication, I will delegate authentication flow to Firebase.
- **Commons lang3**. I will use it to implement equals, hashcode, toString to implement unit tests.

### Describe how you will implement Google Play Services.

Google Play services location API. Using this API, the app will request the last known location of the user's device to register home address.

Firebase Cloud Messaging to receive notification of orders to deliver and stock changes.

Google Maps. To convert coordinates into address. To place markers for the orders to deliver.

## Next Steps: Required Tasks

### Task 1: Configure central data base

- Enable SQL feature to Google Cloud Project
- Create required tables for tortillas delivery, bikers, branches

## Task 2: Google cloud endpoint

- Add database integration. Add Connection settings.
- Add Firebse Cloud Messaging support.
- Create tortillas REST resource.
    - Create tortillas repository to persist info in the database
    - Create operation to retrieve the stock by branch. Tortillas in bikes, orders to deliver.
    - Create operation to retrieve orders by biker
    - Create operation to register the stock of biker. Every time the biker returns to the branch to take more tortillas, it will add a new registry with the initial stock
    - Create operation to register a new order
        - Send push notification to a biker when a new order is received
        - Send push notification to the branch to notify new order
    - Create operation to update an order
        - Send push notification to the branch to notify an order is delivered and paid
- Create bikers REST resource.
    - Create bikers repository to persist info in the database
    - Create operation to register a biker
    - Create operation to update the info of a biker. Registration token.
    - Create operation to remove a biker.
    - 
- Create branches REST resource.
    - Create branches repository to persist info in the database
    - Create operation to register a branch

## Task 3: Client Project Setup

- Configure modules and applications. QuieroTortillas, LaBuena, LaBuenaBikeDriver
- Configure dependencies
- Configure Release Signing
- Configure Dagger dependency injection for applications
- Add Crash reporting support
- Add application theme, and base styles for toolbar, buttons, text, titles, colors.

## Task 4: QuieroTortillas

- User Login
    - Add Login model
    - Create Login view
    - Add Login View Presenter
    - Integrate email and password Firebase authentication
    - Integrate Google Sign-In Firebase

    - Integrate Facebook Sign-In Firebase
    - Add login unit tests
- Forgot Password
    - Create view
    - Create presenter
    - Integrate with Firebase to send the password reset email
    - Add unit tests
- Register Client
    - Create model
    - Create View
    - Create presenter
    - Integrate with Firebase to create an user with email and password account
    - Add unit tests
- Request Tortillas
    - Create model
    - Create presenter
    - Create view
    - Get the last known location for the user
    - Send tortillas request to the server
    - Display notification after request success
    - Add unit tests

## Task 5: LaBuenaBiker

- Set up Firebase Cloud Messaging
- Send registration token to the server
- Update driver notification in central server
    - Create Job Scheduller to send last known location of the driver every 15 minutes
    - Get last known location of the driver
    - Send location to the server
- Create wearable notification
    - Add database to keep orders to deliver
    - Add IntentService to synchronize info from server after notification received.
    - Create DeliverOrders repository to save info
- Biker Login
    - Create model
    - Create view
    - Create presenter
    - Integrate email and password Firebase authentication
    - Add unit tests
- Orders to deliver
    - Create model
    - Create view

  - Create presenter
  - Update DeliverOrders repository to query delivery orders
  - Display custom markers for orders to deliver
  - Add unit tests
- Confirm order as delivered
  - Create model
  - Create View
  - Create Presenter
  - Query pending orders to synchronize to the server.
  - Send deliver order(s) to the server.
  - Update DeliverOrders repository to mark an order as synchronized after successfull send to the server
  - Add tests

## Task 6: LaBuena

- Set up Firebase Cloud Messaging
- Send registration token to the server
- Add local database to keep orders and bikes info
- Add IntentService to synchronize info from server after notification received.
- Request widget update after getting new changes
- Current Stock widget
  - Create view
  - Create widget provider
  - Add tests
- Branch Login
  - Create model
  - Create view
  - Create presenter
  - Integrate email and password Firebase authentication
  - Add unit tests
- Bike Drivers
  - Create model
  - Create fragment view
  - Add adapter view
  - Create presenter
  - Add repository to query the bike drivers
  - Add unit tests
- New Driver
  - Create model
  - Create view
  - Create presenter
  - Integrate with Firebase to create an user with email and password account
  - Send the created driver info to the server

    - Add unit tests
- Edit Driver
    - Create model
    - Create view
    - Create presenter
    - Send updated driver info to the server
    - Integrate with Firebase to send the password reset email
    - Add unit tests